Coursework 2

Exploratory Data Analysis with R

MSc Applied Data Science

Module: Programming with Data (EL4013)

School of Engineering, University of Central Lancashire

March 06, 2021

Table of Content

1. Introduction and Background of EDA packages

The use of predictive models for data analysis has become extremely popular in recent years. The most important and time-consuming step in building these models is to understand the dataset, identify patterns and missing values, and find out the relationship between variables in the dataset. To speed up this process, a large number of exploratory data analysis packages are developed.

The main objective of exploratory data analysis packages is to work with full dataset and not only speed up the Data understanding and data preparation phase, but also keep it simple for the user. The automatic EDA packages have built-in functions which help us to visualise the statistics and numerical information about the datasets. They also provide functions to explain the relationship of different variables. The rise in the development of the EDA packages is a great achievement towards the data analysis process.

2. Summary/Comparison of Selected EDA packages

In this coursework we focussed on four selected EDA packages i.e. funModeling, SmartEDA, DataExplorer and visdat. The selected EDA packages are applied on different datasets in different ways in order to compare their strengths and weaknesses.

2.1. EDA Task Handling Capability

The comparison of each EDA package on basis of their ability to perform different EDA tasks is described in the sections below.

2.1.1. Summary of dataset

All EDA packages provide the summary of dataset by giving the information about variable type, missing values and statistical information in the dataset. FunModeling (using df_status) and DataExplorer (using introduce) give a good overview of full dataset as shown in figure 1 and figure 2 respectively. The visdat package provides the most accurate summary for a small dataset however, it is not reliable for providing summary for larger datasets.

	variable	q_zeros	p_zeros	q_na	p_na	q_inf	p_inf	type	unique
1	tBodyAcc-mean()-X	0	0	0	0	0	0	numeric	7347
2	tBodyAcc-mean()-Y	0	0	0	0	0	0	numeric	7352
3	tBodyAcc-mean()-Z	0	0	0	0	0	0	numeric	7349
4	tGravityAcc-mean()-X	0	0	0	0	0	0	numeric	7351
5	tGravityAcc-mean()-Y	0	0	0	0	0	0	numeric	7352
6	tGravityAcc-mean()-Z	0	0	0	0	0	0	numeric	7352
7	tBodyAccJerk-mean()-X	0	0	0	0	0	0	numeric	7352
8	tBodyAccJerk-mean()-Y	0	0	0	0	0	0	numeric	7352
9	tBodyAccJerk-mean()-Z	0	0	0	0	0	0	numeric	7352
10	tBodyGyro-mean()-X	0	0	0	0	0	0	numeric	7352

Figure 1. Summary of dataset using funModeling

```
   rows columns discrete_columns continuous_colu~ all_missing_col~
  <int>   <int>            <int>            <int>            <int>
1  7352      54                1               53                0
# ... with 4 more variables: total_missing_values <int>, complete_rows <int>,
```

Figure 2. Summary of dataset using DataExplorer

2.1.2. Correlation Plots

The DataExplorer (using plot_correlation) and visdat (using vis_cor) provide correlation function to find out the relationship of different variables. The DataExplorer is more reliable as it converts the categorical values to numerical range and plots their correlation with other variables. Visdat package does not deal with categorical values and only needs numerical columns for this purpose. In additions to that, the correlation table from DataExplorer gives exact values of correlations (figure 3) whereas visdat provides the coloured plot indicating correlation range (figure 4). However, the DataExplorer package needs a clean dataset to perform well. The funModeling lacks correlation table feature however, it provides the correlation of variables with a defined target variable (figure 5).

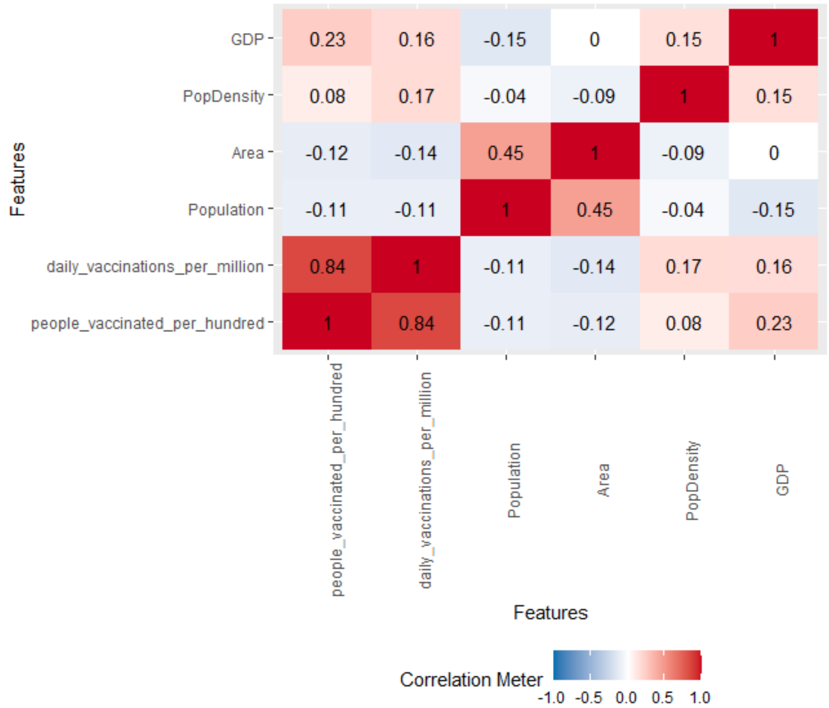

Figure 3. Correlation matrix using DataExplorer

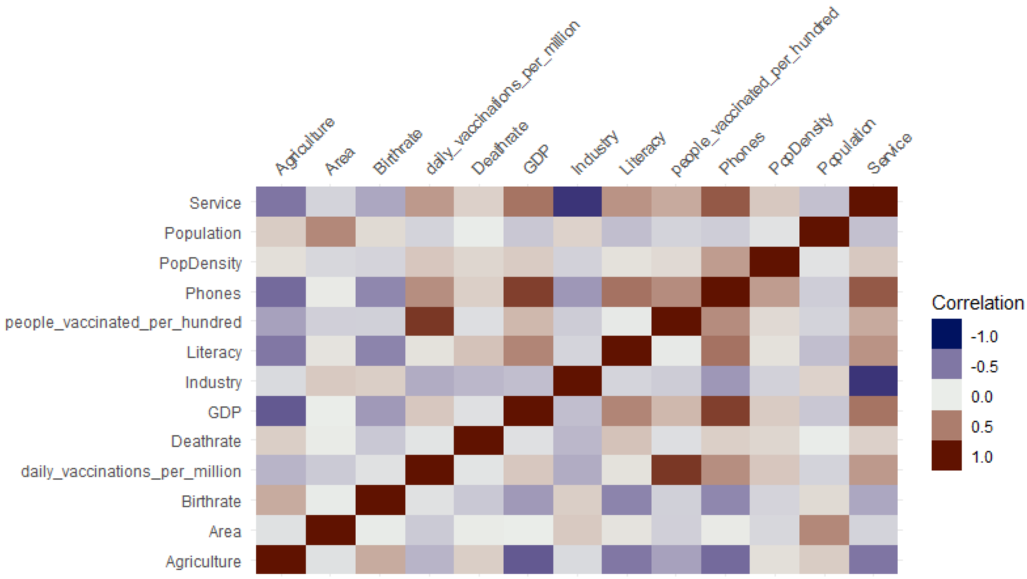

Figure 4. Correlation matrix using visdat

```
                     variable people_vaccinated_per_hundred
1   people_vaccinated_per_hundred                          1.00
2  daily_vaccinations_per_million                          0.77
3                          Phones                          0.35
4                             GDP                          0.26
5                         Service                          0.22
6                      PopDensity                          0.05
7                        Literacy                         -0.01
8                        Industry                         -0.04
9                       Birthrate                         -0.08
10                     Population                         -0.10
11                           Area                         -0.11
12                      Deathrate                         -0.14
13                    Agriculture                         -0.31
```

Figure 5. Correlation of variables using funModeling

2.1.3. Data Cleaning

The data manipulation of a dataset is only limited to a few packages. FunModeling and DataExplorer provide functions to normalise the data, create dummy variables and transform the columns . The funModeling has advantage in this comparison as it also provides function (discretize_df) to discretise the data between two limits. The visdat and SmartEDA lack the feature of data manipulation.

2.1.4. Generating Reports

The SmartEDA and DataExplorer have ability to generate report and save it as a file by using a single function (ExpReport and create_report respectively). The report contains the summary and plots sorted by the type of variables. FunModeling lacks the function of producing reports however, it provides a function to save the outputs and graphs in a file. The reporting feature of SmartEDA and DataExplorer make them more reliable compared to FunModeling and visdat.

2.1.5. Information about missing values

Visdat provides a variety of functions to visualise the missing values which makes it more reliable compared to other packages.

2.2. Versatile

The funModeling is the most versatile package out of all four packages. It covers a wide range of tasks which other libraries do not provide (e.g. discretize_df etc). It is the only package which provide functions to visualise the predictive models. Figure 6 shows the performance matrix obtained using funModeling.

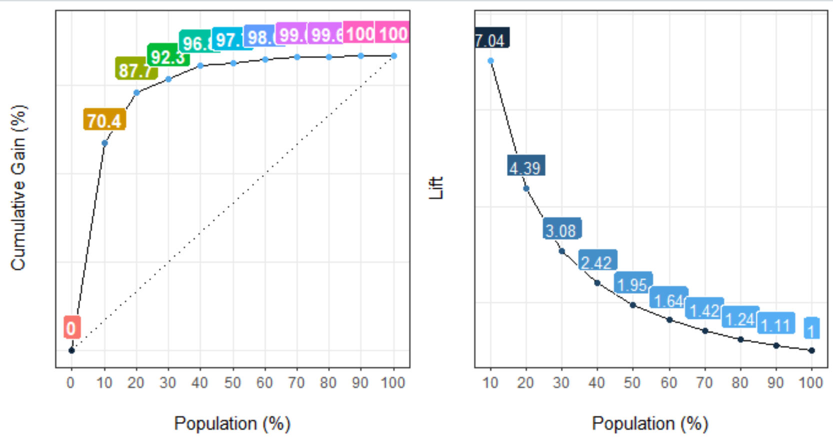

Figure 6. Cumulative Gain and Lift Curve using funModeling

2.3. Performance

The performance of all EDA packages is more or less similar. FunModeling package, however, performs a lot slower while plotting boxplots compared to SmartEDA.

2.4. User Experience

In terms of user experience, the SmartEDA is the best out of all packages under consideration. It chooses the variables automatically and performs the statistical analysis and provides information about both numeric and non-numeric variables. There is no effort needed to prepare the report while using SmartEDA as it provides a function to do so.

2.5. Learning Difficulties for new Users

The funModeling package is the easiest to use out of all other packages under consideration. It takes the whole dataset as input and gives the output. It automatically ignores the categorical values where needed (i.e. while performing correlation). DataExplorer needs more effort from the user to clean the dataset before applying functions.

2.6. Manual Quality

The manual quality of all packages is quite good as they all give information about defined functions, their usage and description. The examples of how to use them in codes is also given in each manual. The descriptions of funModeling and visdat are more powerful compared to other packages as they also explain when to use the functions and the limitations of each function.

2.7. Data Size Limit

The funModeling, DataExplorer and SmartEDA give good insights for a dataset with medium range of features. Visdat is not suitable for a dataset with a higher number of variables.

2.8. Computational Cost

Computation cost of funModeling is the most out of all four packages. It takes a lot of processing time while plotting boxplots which immensely increases the computational cost.

2.9. Platform Requirements

There are no specific platform requirements for any of the packages under considerations. All packages can be easily used using RStudio without needing any specific machine requirements.

3. Case Studies

3.1. Case Study 1: Human Activity Recognition

In this case study Human Activity Recognition dataset is used to analyse and predict the human activity using smartphones. Different libraries i.e. funModeling, SmartEDA and visdat are used to perform the exploratory data analysis to find out which variables can distinguish the human activities and results are compared to indicate their strengths and weaknesses.

The dataset consists of 563 features measured using smartphone sensors; however, it is not possible to analyse all of them hence only mean variables are used to carry out the analysis. This reduced the number to 54 variables.

```
human_df <- data %>% select(contains("mean") |
                             contains("Mean") |
                             contains("Activity"))
```

3.1.1. SmartEDA

A clear and concise overview of the data is seen (figure 7) while using SmartEDA which helps to understand the dimensions of the dataset, variable names, overall missing summary, and data types of all variables. The structure of data is seen in figure 8.

```
ExpData(data=human_df,type=1)
```

	Descriptions	Value
1	Sample size (nrow)	7352
2	No. of variables (ncol)	54
3	No. of numeric/interger variables	53
4	No. of factor variables	0
5	No. of text variables	1
6	No. of logical variables	0
7	No. of identifier variables	31
8	No. of date variables	0
9	No. of zero variance variables (uniform)	0
10	%. of variables having complete cases	100% (54)
11	%. of variables having >0% and <50% missing cases	0% (0)
12	%. of variables having >=50% and <90% missing cases	0% (0)
13	%. of variables having >=90% missing cases	0% (0)

Figure 7. Summary of data using SmartEDA

```
ExpData(data=human_df,type=2)
```

	Index	Variable_Name	Variable_Type	Per_of_Missing
1	1	tBodyAcc-mean()-X	numeric	0
2	2	tBodyAcc-mean()-Y	numeric	0
3	3	tBodyAcc-mean()-Z	numeric	0
4	4	tGravityAcc-mean()-X	numeric	0
5	5	tGravityAcc-mean()-Y	numeric	0
6	6	tGravityAcc-mean()-Z	numeric	0
7	7	tBodyAccJerk-mean()-X	numeric	0
8	8	tBodyAccJerk-mean()-Y	numeric	0
9	9	tBodyAccJerk-mean()-Z	numeric	0
10	10	tBodyGyro-mean()-X	numeric	0

Figure 8. Structure of data using SmartEDA

SmartEDA does not provide any function to compute correlation of variables hence the analysis is carried out by comparing each variable with output variable one by one. This requires quite a lot of time and effort to get results for a dataset consisting of large number of features. Figure 9 shows that gravity acceleration mean best differentiates if the body is laying, sitting, or standing.

```
ExpNumViz(human_df[,c("tGravityAcc-mean()-Y","Activity")],
          target="Activity",
          nlim=40,
          fname=NULL,
          col=NULL,
          Page=NULL,
          sample=1)
```

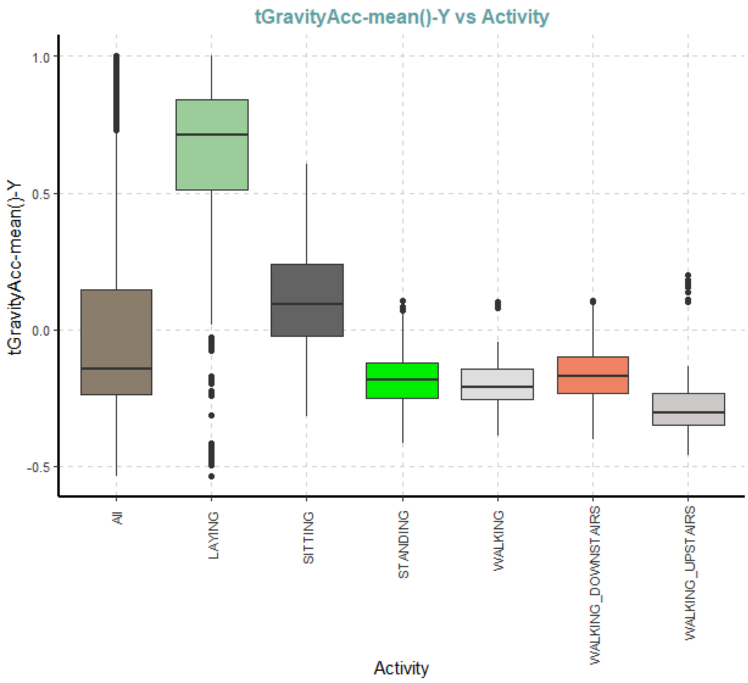

Figure 9. Boxplot of Activity vs Gravity Acceleration Mean using SmartEDA

The body acceleration magnitude mean is used to differentiate if the person is walking upstairs or downstairs as shown in figure 10.

```
ExpNumViz(human_df[,c("tBodyAccMag-mean()","Activity")]
          target="Activity",
          nlim=40,
          fname=NULL,
          col=NULL,
          Page=NULL,
          sample=1)
```

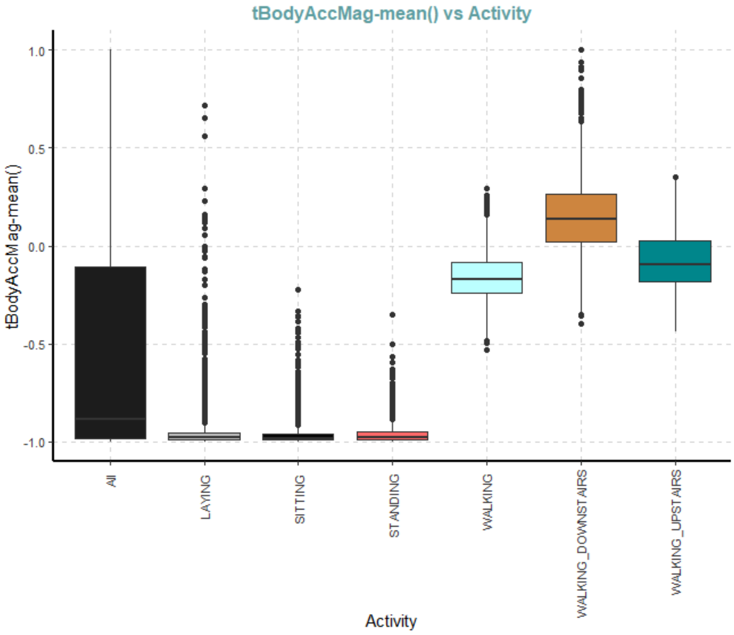

Figure 10. Boxplot of Activity vs Acceleration Magnitude Mean using SmartEDA

From the above plots, it is concluded that:

- If tGravityAcc-mean()-Y > 0.5 -> Laying
- If 0 < tGravityAcc-mean()-Y < 0.3 -> Sitting
- If tGravityAcc-mean()-Y < 0 -> Standing
- If tBodyAccMag-mean() > 0 -> Walking downstairs
- If tBodyAccMag-mean < 0 -> Walking upstairs

3.1.2. FunModeling

funModeling gives a brief overview and statistical analysis of dataset as shown in figure 11 and figure 12 respectively.

```
df_status(human_df)
```

	variable	q_zeros	p_zeros	q_na	p_na	q_inf	p_inf	type	unique
1	tBodyAcc-mean()-X	0	0	0	0	0	0	numeric	7347
2	tBodyAcc-mean()-Y	0	0	0	0	0	0	numeric	7352
3	tBodyAcc-mean()-Z	0	0	0	0	0	0	numeric	7349
4	tGravityAcc-mean()-X	0	0	0	0	0	0	numeric	7351
5	tGravityAcc-mean()-Y	0	0	0	0	0	0	numeric	7352
6	tGravityAcc-mean()-Z	0	0	0	0	0	0	numeric	7352
7	tBodyAccJerk-mean()-X	0	0	0	0	0	0	numeric	7352
8	tBodyAccJerk-mean()-Y	0	0	0	0	0	0	numeric	7352
9	tBodyAccJerk-mean()-Z	0	0	0	0	0	0	numeric	7352
10	tBodyGyro-mean()-X	0	0	0	0	0	0	numeric	7352
11	tBodyGyro-mean()-Y	0	0	0	0	0	0	numeric	7352

Figure 11. Overview of dataset using funModeling

```
profiling_num(human_df)
```

	variable	mean	std_dev	variation_coef	p_01
1	tBodyAcc-mean()-X	0.274488125	0.07026133	0.2559722	0.07720714
2	tBodyAcc-mean()-Y	-0.017695427	0.04081052	-2.3062752	-0.10321380
3	tBodyAcc-mean()-Z	-0.109141020	0.05663519	-0.5189175	-0.25572589
4	tGravityAcc-mean()-X	0.664121889	0.51676314	0.7781149	-0.70721338
5	tGravityAcc-mean()-Y	0.011006040	0.37223721	33.8211739	-0.43341381
6	tGravityAcc-mean()-Z	0.093920274	0.34779248	3.7030608	-0.50877105
7	tBodyAccJerk-mean()-X	0.079125881	0.18092015	2.2864851	-0.47864888
8	tBodyAccJerk-mean()-Y	0.008555454	0.16370678	19.1347851	-0.49620589
9	tBodyAccJerk-mean()-Z	-0.004693134	0.15990963	-34.0731034	-0.49642518

Figure 12. Statistical Analysis of dataset using funModeling

The frequency distribution of variables is analysed using funModeling in order to find out the amount of data corresponding to each activity. Figure 13 shows that there is a reasonable distribution of data present in each activity.

```
freq(human_df$Activity)
```

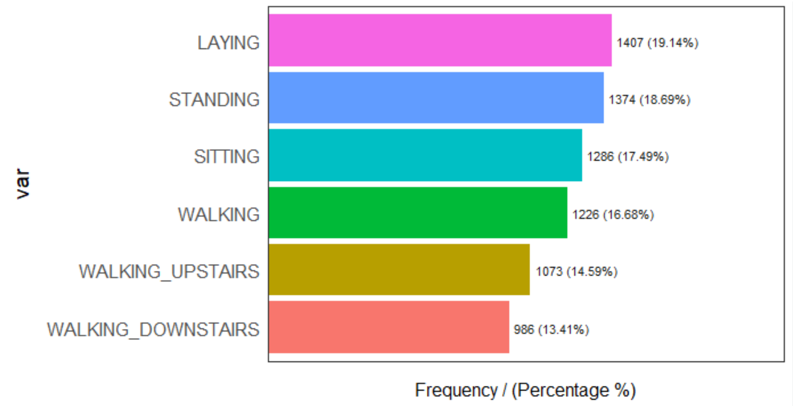

Figure 13. Frequency Distribution of Activity using funModeling

The correlation of each variables with target variable is computed to see which variables contribute the most in deciding the activity. Figure 14 and figure 15 show the correlation of top few variables with positive and negative correlation respectively. The body acceleration mean, and gravity acceleration mean have highest positive and negative correlation hence they can be used to distinguish different activities.

```
cor <- correlation_table(human_df1, 'Activity')
```

```
                 Variable Activity
1                Activity     1.00
2       tBodyAccMag-mean()    0.83
3    tGravityAccMag-mean()    0.83
4      tBodyGyroMag-mean()    0.83
5        fBodyAcc-mean()-Y    0.81
6        fBodyAcc-mean()-X    0.80
7       fBodyGyro-mean()-Z    0.80
8       fBodyAccMag-mean()    0.79
```

Figure 14. Top variables showing with positive correlation

```
27   fBodyBodyAccJerkMag-meanFreq()   -0.37
28           fBodyAcc-meanFreq()-X    -0.38
29           fBodyAcc-meanFreq()-Z    -0.38
30       fBodyAccJerk-meanFreq()-Y    -0.61
31       fBodyAccJerk-meanFreq()-Z    -0.62
32            angle(X,gravityMean)    -0.62
33            tGravityAcc-mean()-Z    -0.63
34       fBodyAccJerk-meanFreq()-X    -0.67
35            tGravityAcc-mean()-Y    -0.75
```

Figure 15. Top variables showing with negative correlation

Plotar function of funModeling is used to boxplot acceleration magnitude mean vs activity and gravity acceleration vs activity. The results obtained from these plots are same as those obtained from SmartEDA (already discussed in SmartEDA section) as shown in figure 9 and figure 10. However, the processing time of getting results from this function is very high which increases the computational cost.

3.1.3. Visdat

Visdat provides good functions to view and deal with missing values. Figure 16 shows that the data does not contain any missing values.

```
vis_expect(human_df, ~.x %in% c("N/A", "NA"))
```

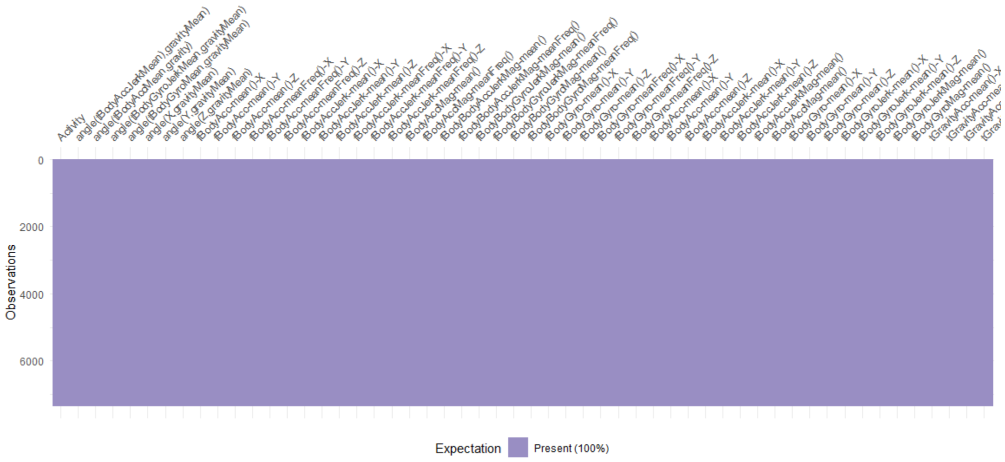

Figure 16. An overview of missing and complete data

To analyse the frequency distribution of activities, each activity is separately checked as visdat does not deal with the whole variable at once. Figure 17 shows the amount of data corresponding to sitting activity.

```
vis_expect(human_df[,563:563], ~.x =="SITTING")
vis_expect(human_df[,563:563], ~.x =="STANDING")
vis_expect(human_df[,563:563], ~.x =="LAYING")
vis_expect(human_df[,563:563], ~.x =="WALKING")
```

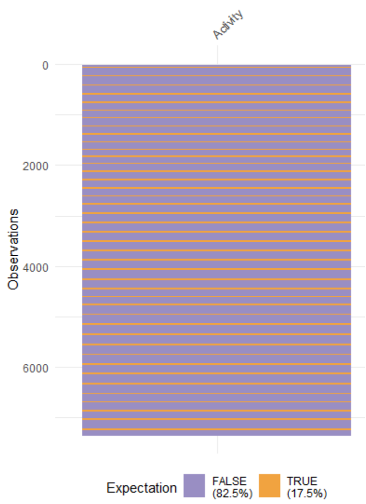

Figure 17. Frequency distribution corresponding to Sitting Activity

The correlation of a variables is found out and analysed which variables contribute more. Figure 18 shows the correlation of 20 variables since it is impossible to analyse all 54 variables at once.

```
vis_cor(human_df1)
```

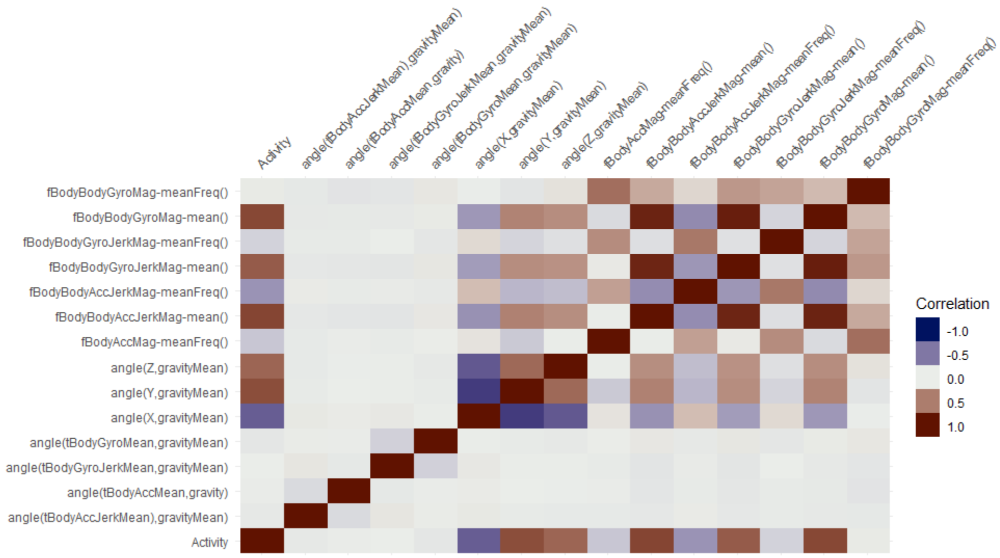

Figure 18. Correlation of variables using visdat

The exact values of correlation cannot be determined from visdat plot however, the body acceleration mean, and gravity acceleration mean have high positive and negative correlations respectively and they can be used to distinguish different activities.

Visdat does not provide function for boxplot hence we cannot visualise the boundaries for each activity using this package.

Discussion

From the above case study, it is noted that funModeling, DataExplorer and SmartEDA a good overview of data whereas the visdat gives minimum information about the data.

The distribution of variables in a dataset is well clarified by funModeling function whereas visdat needs more time and computational cost to give information about the distribution.

Visdat gives better information about missing values and correlation of variables out of all packages under consideration. FunModeling only provides correlation with a target variable and SmartEDA lacks the correlation function.

SmartEDA performs the best boxplots, funModeling also provides boxplots function but the processing time is high compared to SmartEDA.

3.2. Case Study 2: Covid-19 Vaccination Progress

In this case study funModeling and DataExplorer packages are used to find out the factors affecting the vaccination process in a country. The two datasets containing vaccination data and countries of the world are combined to make a new dataset which is used to find the factors.

The data is prepared, and correlations are analysed using two libraries to predict the factors. The results are then compared to find out which library is more effective while performing correlations.

It is discussed earlier that visdat function does not deal with categorial values hence categorical columns are ignored. Figure 19 and figure 20 show the correlation of variables among each other using visdat and DataExplorer respectively.

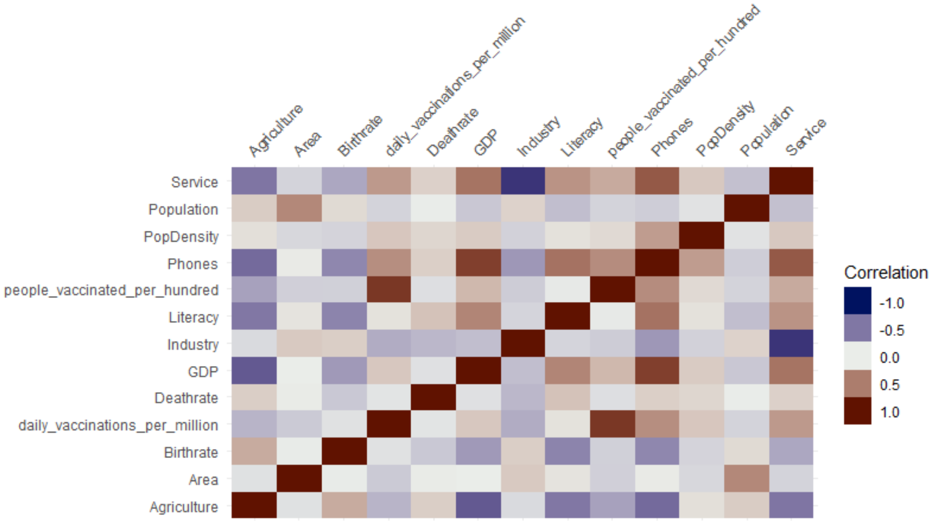

Figure 19. Correlation of variables using funModeling

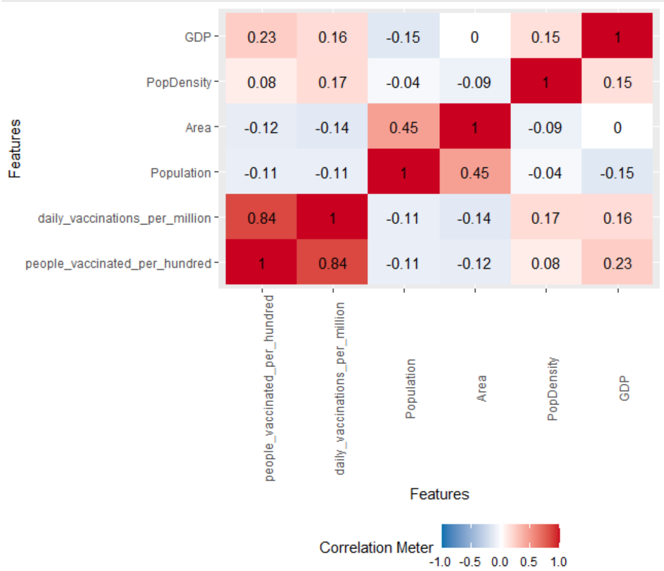

Figure 20. Correlation of variables using DataExplorer

Both correlation tables show that the percentage of people vaccinated in a country mainly depend on Phones, services, and GDP. The number of phones in a country refer to more information as well as this number is directly related to the literacy rate. This explains that countries with good information are more likely to get more vaccinations. GDP and services also greatly affect the vaccination process in a country as developed countries with more service facilities can afford and provide vaccinations better than that of developing and undeveloped countries. It is interesting to know that countries which are more relying on agriculture are less progressive in this process compared to countries relying more on industry.

Discussion

Visdat gives a clear graph which is simple and quick to understand the correlations. The high correlation variables can be quickly analysed by searching for colours with high correlation. The plot using DataExplorer mentions the exact correlation of variables which can by time-consuming while finding high correlation variables. Sometimes the exact correlation values using DataExplorer can be helpful however, it only works efficiently for a clean dataset.

3.3. Case Study 3: Bankruptcy Prediction (Using funModeling)

In this case study, the prediction of bankruptcy of a company is analysed using funModeling package. As discussed earlier, funModeling provides feature to visualise the predictive model, hence funModeling is used to predict the bankruptcy of the company and visualise the score of the model using gain and lift curves.

The dataset contains data of different companies upon which the bankruptcy is based on. It consists of 96 columns containing different stock exchange features. The complete summary of dataset is shown in figure 21.

```
df_status(data)
```

	variable	q_zeros	p_zeros	q_na	p_na	q_inf	p_inf	type	unique
1	Bankrupt?	6599	96.77	0	0	0	0	numeric	2
2	ROA(C) before interest and depreciation before interest	1	0.01	0	0	0	0	numeric	3333
3	ROA(A) before interest and % after tax	1	0.01	0	0	0	0	numeric	3151
4	ROA(B) before interest and depreciation after tax	1	0.01	0	0	0	0	numeric	3160
5	Operating Gross Margin	1	0.01	0	0	0	0	numeric	3781
6	Realized Sales Gross Margin	1	0.01	0	0	0	0	numeric	3788
7	Operating Profit Rate	1	0.01	0	0	0	0	numeric	3376
8	Pre-tax net Interest Rate	1	0.01	0	0	0	0	numeric	3789
9	After-tax net Interest Rate	1	0.01	0	0	0	0	numeric	3604
10	Non-industry income and expenditure/revenue	1	0.01	0	0	0	0	numeric	2551
11	Continuous interest rate (after tax)	1	0.01	0	0	0	0	numeric	3617
12	Operating Expense Rate	1	0.01	0	0	0	0	numeric	2966

Figure 21. Overview of dataset using funModeling

The correlation of different variables with ‘Bankrupt’ variable is computed after visualisation and preparation as shown in figure 22. There are 94 variables correlating with Bankruptcy, however, variables with high correlation are considered. It can be noticed that debt ratio has the highest positive correlation whereas the Net Income to Total assets has the highest negative correlation with bankruptcy.

```
cor <- correlation_table(data, 'Bankrupt')
filter(cor, cor$Bankrupt >0.15 | cor$Bankrupt < -0.15)
```

	Variable	Bankrupt
1	Bankrupt	1.00
2	Debt ratio %	0.25
3	Current Liability to Assets	0.19
4	Borrowing dependency	0.18
5	Current Liability to Current Assets	0.17
6	Liability to Equity	0.17
7	Net Value Per Share (C)	-0.16
8	Net Value Per Share (B)	-0.17
9	Net Value Per Share (A)	-0.17
10	Net Income to Stockholder's Equity	-0.18
11	Working Capital to Total Assets	-0.19
12	Per Share Net profit before tax (Yuan ¥)	-0.20
13	Net profit before tax/Paid-in capital	-0.21
14	Persistent EPS in the Last Four Seasons	-0.22
15	Retained Earnings to Total Assets	-0.22
16	Net worth/Assets	-0.25
17	ROA(C) before interest and depreciation before interest	-0.26
18	ROA(B) before interest and depreciation after tax	-0.27
19	ROA(A) before interest and % after tax	-0.28
20	Net Income to Total Assets	-0.32

Figure 22. Correlation of top few variables using funModeling

The columns which greatly affect the bankruptcy of a company are used to create the machine learning algorithm using funModeling and the score is added to a new column. This score is then used to in gain_lift function to calculate the performance metrics.

```
fit_glm=glm(Bankrupt ~ data$`Debt ratio %` + data$`Net Income to Total Assets`,
            data=data, family = binomial)

data$score=predict(fit_glm, newdata=data, type='response')

gain_lift(data, score='score', target='Bankrupt')
```

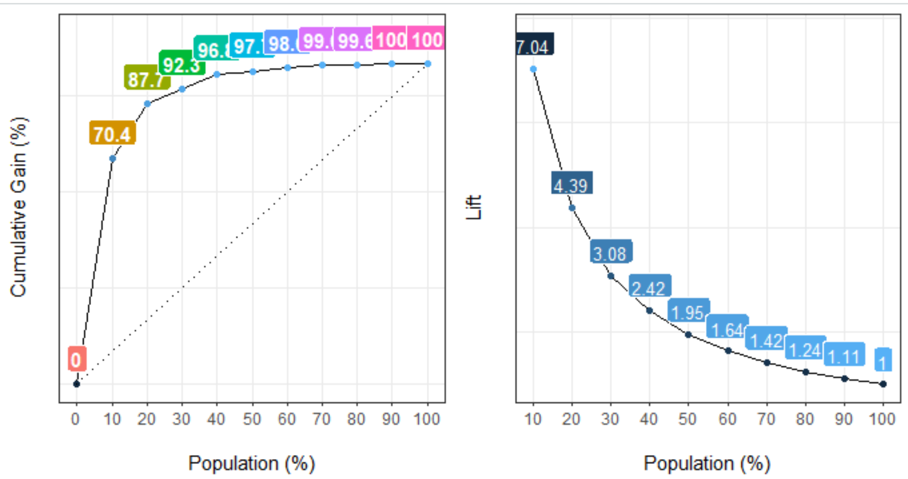

Figure 23. Cumulative Gain and Lift Curves using funModeling

Figure 23 shows the performance metrics obtained using funModeling. The cumulative gain and lift curves explain how well the classification model is predicting. Higher values of gain at the beginning of the population explain that the model is separating classes very well. The predictive model reaches 70% of positive cases only around 10% of population, which represents that the model has captured information from the data very well. This score can further be improved by introducing more columns with high correlation with the target.

Discussion

FunModeling can be efficiently used for visualisation of predictive models. This is an advantage while using funModeling since no other package provides this function.

4. Conclusion

From the above case studies, it can be concluded that SmartEDA gives clear and concise summary of data. The distribution of variables in a dataset is well clarified by funModeling function. Visdat gives better information about missing values and correlation of variables. SmartEDA gives the best boxplots with minimum computational cost whereas funModeling does the same job with high computational cost.

Visdat gives a clear graph of correlations which is simple and quick to understand. The DataExplorer mentions the exact correlation values which can be helpful but more time consuming while analysing. FunModeling is the most versatile package which also provides feature to visualise the score of predictive models using gain and lift curves.

It is impossible to decide which package is the best as all packages have their pros and cons. The selection of package depends on what features are going to be visualised by the user. FunModeling is the best package to choose because of its ability to deal with multiple tasks however, the processing time and computational cost has to be compromised for some functions. For a clean dataset, DataExplorer is the best choice. For a dataset with less features, visdat can be the best choice as it provides the most accurate information. SmartEDA can be chosen on basis of its ability to provide clear and concise boxplots with minimum computational cost.

5. References

Staniak, M. and Biecek, P., 2019. The Landscape of R Packages for Automated Exploratory Data Analysis. *arXiv preprint arXiv:1904.02101*.

D. Ubrangala, K. Rama, and R. Kondapalli. SmartEDA: Summarize and Explore the Data, 2018. URL https://CRAN.R-project.org/package=SmartEDA. R package version 0.3.0.

P. Casas. funModeling: Exploratory Data Analysis and Data Preparation Tool-Box Book, 2019. URL https://CRAN.R-project.org/package=funModeling. R package version 1.

B. Cui. DataExplorer: Automate Data Exploration and Treatment, 2019. URL https://CRAN.R-project.org/package=DataExplorer. R package version 0.8.0.

S. Putatunda, K. Rama, D. Ubrangala, and R. Kondapalli. SmartEDA: An R Package for Automated Exploratory Data Analysis. arXiv e-prints, art. arXiv:1903.04754, Mar 2019.

N. Tierney. visdat: Visualising whole data frames. JOSS, 2(16):355, 2017. doi: 10.21105/joss.00355.

Appendix

Case Study 1

```
library(tidyverse)
library(dplyr)

#Read the dataset and analuyse the dimensions and head
data <- read_csv('C:/MSc/Project Work/Human Activity/train.csv')

dim(data)

human_df <- data %>% select(contains("mean") |
                              contains("Mean") |
                              contains("Activity"))

# Create another dataframe
human_df1 <- human_df

human_df1$Activity <- as.character(human_df1$Activity)
human_df1[human_df1=="LAYING"] <- '1'
human_df1[human_df1=="SITTING"] <- '2'
human_df1[human_df1=="STANDING"] <- '3'
human_df1[human_df1=="WALKING"] <- '4'
human_df1[human_df1=="WALKING_DOWNSTAIRS"] <- '5'
human_df1[human_df1=="WALKING_UPSTAIRS"] <- '6'
human_df1$Activity <- as.numeric(human_df1$Activity)

#----------------------------Using FunModelling--------------------------------

library(funModeling)

# First of all,let's analyse the summary of the dataset
df_status(human_df)

# Plot the numerical variables
plot_num(human_df)

# Let us view the statistics of data
profiling_num(human_df)

# Now we will analyse the frequency distribution of variables
# Let's analyse how much data is there in dataframe for each activity
freq(human_df$Activity)

# The distribution of variables is reasonable now let's analyse the datasets.
# Let's view the correlation of all values that affect activity
cor <- correlation_table(human_df1, 'Activity')

filter(cor,cor$Activity>0.7)
filter(cor,cor$Activity<0.7)

plotar(data=human_df,
       input = human_df$`tBodyAccMag-mean()`,
       target= "Activity",
       plot_type="boxplot")

plotar(data=human_df,
       input = human_df$`fBodyAccJerk-entropy()-X`,
       target= "Activity",
       plot_type="boxplot")

plotar(data=human_df,
       input = human_df$`fBodyAccJerk-entropy()-X`,
       target= "Activity",
       plot_type="boxplot")
```

```
plotar(data=human_df,
       input = human_df$`fBodyAcc-mean()-Y`,
       target= "Activity",
       plot_type="boxplot")

#-----------------------------SmartEDA----------------------------

library("SmartEDA")

# Now let's use SmartEDA to Overview the data
# understand the dimensions of the dataset, variable names,
# overall missing summary and data types of each variables.
ExpData(data=human_df,type=1)

# Now let's see the structure of data
ExpData(data=human_df,type=2)


# Now find out what factors are needed to classify the
ExpNumViz(human_df[,c("tBodyAccMag-mean()","Activity")],
          target="Activity",
          nlim=40,
          fname=NULL,
          col=NULL,
          Page=NULL,
          sample=1)

ExpNumViz(human_df[,c("tBodyAcc-sma()","Activity")],
          target="Activity",
          nlim=40,
          fname=NULL,
          col=NULL,
          Page=NULL,
          sample=1)

ExpNumViz(human_df[,c("angle(X,gravityMean)","Activity")],
          target="Activity",
          nlim=40,
          fname=NULL,
          col=NULL,
          Page=NULL,
          sample=1)

ExpNumViz(human_df[,c("tGravityAcc-mean()-Y","Activity")],
          target="Activity",
          nlim=40,
          fname=NULL,
          col=NULL,
          Page=NULL,
          sample=1)

#-------------------------------Visdat------------------------------------

library(visdat)

# Check the overview of data
glimpse(data)

# Check the data missing in the dataframe
vis_miss(human_df, warn_large_data = FALSE)

# It warns you about large data
# The function gives information about missing values efficiently
# The computational time is high for large dataset.
```

```
# vis_compare function compared the two dataframes and tells us what has
# changed in the dataframe
# We know that we have changed the values of activity to number in human_df1.
# Let's check the difference

vis_compare(human_df[,557:563], human_df1[,557:563])

# Let's check if there are any missing values
vis_expect(human_df, ~.x %in% c("N/A", "NA"))

# No  missing values since 100% data does not containg NA or N/A

#Let's check what percentage of data is for Standing, Sitting, laying and others

vis_expect(human_df[,54:54], ~.x =="SITTING")
vis_expect(human_df[,54:54], ~.x =="STANDING")
vis_expect(human_df[,54:54], ~.x =="LAYING")
vis_expect(human_df[,54:54], ~.x =="WALKING")

# Now let's check the correlation of all variables

vis_cor(human_df1[,1:10])
```

Case Study 2

```
library(tidyverse)
library(dplyr)

#Read the dataset containting vaccination details
covid_vacc <- read_csv('C:/MSc/Programming/Vaccine/country_vaccinations.csv')
count_prof <- read_csv('C:/MSc/Programming/Vaccine/countries of the world.csv')

ExpData(data=vacc_data,type=1)

#Check the dimestions of dataset
dim(covid_vacc)

# Check first few elements of dataset
head(covid_vacc)

# View the full dataset
view(covid_vacc)

# Select the data that we are going to need
vacc_data <- covid_vacc[,c("country",
                           "total_vaccinations",
                           "people_vaccinated",
                           "people_fully_vaccinated",
                           "people_vaccinated_per_hundred",
                           "daily_vaccinations_per_million")]


vacc_data <- covid_vacc[,c("country",
                           "people_vaccinated_per_hundred",
                           "daily_vaccinations_per_million")]


# England, Scotland, Wales and Northern Ireland are part of UK hence drop them
vacc_data <- filter(vacc_data,
                    country!= 'England' &
                      country!= 'Scotland' &
                      country!= 'Wales' &
                      country!= 'Northern Ireland')
```

```
# Select the factors that are going to be observed
count_data <- count_prof[,c("Country",
                            "Population","Area (sq. mi.)",
                            "Pop. Density (per sq. mi.)",
                            "GDP ($ per capita)", "Phones (per 1000)",
                            "Literacy (%)",
                            "Birthrate", "Deathrate", "Agriculture",
                            "Industry", "Service")]

# The data contains some commas instead of dots, let's replace the values
count_data$region <- gsub(" ", "", count_data$region)
count_data$PopDensity <- as.numeric(gsub(",", ".", count_data$PopDensity))
count_data$Phones <- as.numeric(gsub(",", ".", count_data$Phones))
count_data$Literacy <- as.numeric(gsub(",", ".", count_data$Literacy))
count_data$Birthrate  <- as.numeric(gsub(",", ".", count_data$Birthrate))
count_data$Deathrate  <- as.numeric(gsub(",", ".", count_data$Deathrate))
count_data$Agriculture <- as.numeric(gsub(",", ".", count_data$Agriculture))
count_data$Industry <- as.numeric(gsub(",", ".", count_data$Industry))
count_data$Service <- as.numeric(gsub(",", ".",count_data$Service))

# Make another data frame by the total number
vacc_data[is.na(vacc_data)] <- 0

sum_vacc <- vacc_data %>%
  group_by(country) %>%
  arrange(people_vaccinated_per_hundred) %>%
  slice(n())

# Now let us joing both dataframes by selecting the chosen columns only
new_df = merge(sum_vacc,count_data,by='country')



#------------------Let us check the performance of each countries---------------

# Check performance of countries on basis of poeple fully vaccinated
df <- arrange(sum_vacc,desc(people_fully_vaccinated))
barplot(df[1:20,]$people_fully_vaccinated,
        names.arg = df[1:20,]$country,
        col = 'blue', las =2)

# Check percentage of people fully vaccinated on basis of total population
df <- arrange(sum_vacc,desc(people_vaccinated_per_hundred))
barplot(df[1:20,]$people_vaccinated_per_hundred,
        names.arg = df[1:20,]$country,
        col = 'blue', las =2)

#--------------------------------SmartEDA----------------------------------#

library("SmartEDA")

# Now let's use SmartEDA to Overview the data
# understand the dimensions of the dataset, variable names, overall missing
# summary and data types of each variables.
ExpData(data=new_df,type=1)

# Now let's see the structure of data
ExpData(data=new_df,type=2)
```

```
# Now plot
ExpNumViz(new_df[,c("people_vaccinated_per_hundred","Phones")],
          target="Phones",
          nlim=40,
          fname=NULL,
          col=NULL,
          Page=NULL,
          sample=1)


ExpNumViz(new_df[1:10,],target = "country",
          type = 2,
          nlim = 2,
          col = c("red", "green", "blue"),
          Page = NULL,
          sample = 2,
          scatter = FALSE,
          gtitle = "Box plot: ")

#------------------------------FunModelling--------------------------------#

library(funModeling)

# Check the status of data
status(new_df)

# Plot the numerical variables
plot_num(new_df)

# Let us view the statistics of data
profiling_num(new_df)

# Let us analyse the frequency distribution of variables
# Let's analyse how much data is from each country
freq(new_df$Country)
# We have equal data from 10 countries

# Let us analyse how many males have covid compared to females.
freq(new_df$Gender_Male)
# 33 % males and 67 % females

# Let's analyse what percentage has no symptoms
freq(new_df$None_Sympton)
# There are 5.25% people who have np symptoms

# Lets analyse the correlation


correlation_table(new_df, "people_vaccinated_per_hundred")

var_rank_info(new_df, "people_vaccinated_per_hundred")

# ----------------------------Visdat-------------------------------------
library(visdat)

glimpse(new_df)

# Check the data missing in the dataframe
vis_miss(new_df, warn_large_data = FALSE)

# It warns you about large data
# The function gives information about missing values efficiently
# The computational time is high for large dataset.
```

```r
# Let's check if there are any missing values
vis_expect(human_df, ~.x %in% c("N/A", "NA"))

# Now let's check the correlation of all variables

vis_cor(new_df[,2:14])

# ----------------------------Data Explorer--------------------------------

library(DataExplorer)

# Let's understand the dataset
introduce(new_df)

# Analyse the details visually
plot_intro(new_df)

# To visualise missing values in each profile
plot_missing(new_df)

# Let's check the frequency of discrete columns
plot_bar(new_df)

# Now let's analyse the frequency distribution of continuous variables
plot_histogram(new_df)

plot_correlation(
  new_df[,2:14],
  type = c("all", "discrete", "continuous"),
  maxcat = 50L,
  cor_args = list(),
  geom_text_args = list(),
  title = NULL,
  ggtheme = theme_gray(),
  theme_config = list(legend.position = "bottom",
                      axis.text.x = element_text(angle = 90)))
```

Case Study 3

```r
library(tidyverse)
library(dplyr)

#Read the dataset and analyse the dimensions and head
data <- read_csv('C:/MSc/Project Work/Bankrupcy/data.csv')

#-------------------------------FunModeling----------------------------------

library(funModeling)

# First of all,let's analyse the summary of the dataset

df_status(data)

# Check the details about data

di=data_integrity(data)

# Let us analyse the distribution of numerical variables
plot_num(data)

# Let us analyse the statistical details about the data
profiling_num(data)
```

```r
# Change name of Bankrupt column as ? gives error with some functions

names(data)[1] <- "Bankrupt"

# Let's analyse the distribution of output variable

freq(data, input = data$`Bankrupt`, plot = TRUE, na.rm = FALSE)

# We can see that the data is strongly unbalanced,
# Let's try to balance it to improve performances.


ban_yes <- filter(data, data$`Bankrupt` == 1)
ban_no <- filter(data, data$`Bankrupt` == 0)

# Now select 220 rows randomly to balance the data
ban_no1 <- sample_n(ban_no, 220)

df <- rbind(ban_yes, ban_no1)

set.seed(42)
dataframe <- df[sample(nrow(df)), ]

#Now let's check the distribution of variables
freq(dataframe, input = dataframe$`Bankrupt`, plot = TRUE, na.rm = FALSE)

# Let's find the correlation
cor <- correlation_table(data, 'Bankrupt')

filter(cor, cor$Bankrupt >0.15 | cor$Bankrupt < -0.15)

cross_plot(data=dataframe, input=c("Debt ratio %",
                                   " Net worth/Assets"),
                                    target="Bankrupt?")


plotar(data=data, input = data$`Debt ratio %`, target= "Bankrupt",
                                               plot_type="boxplot")

d_bins=discretize_get_bins(data=data, input=c("Tax rate (A)", "Debt ratio %"),
                                               n_bins=5)


#Plot of var_rank_info to visualize better
#ggplot(cor, aes(x=reorder(var, gr), y = gr, fill = var)) +
  geom_bar(stat = 'identity') + coord_flip() + theme_bw() + xlab('') +
  ylab('Each feature importance according to the gr (Information Gain)') +
  guides(fill = FALSE)

#-------------------------Predictive Model Performance--------------------------

# Create machine learning model and get its scores for positive case
fit_glm=glm(Bankrupt ~ data$`Debt ratio %` + data$`Net Income to Total Assets`,
            data=data, family = binomial)

data$score=predict(fit_glm, newdata=data, type='response')



# Calculate performance metrics
gain_lift(data, score='score', target='Bankrupt')
```